CLAIMS

1. A method in a computer system for processing of a thread waiting for access to a memory location, the method comprising:

when access by a thread to the memory location is blocked,

enabling an exception to be raised when that memory location is accessed; and

blocking execution of the thread; and

when an exception is raised as a result of access by another thread to that memory location,

completing the access by that other thread to that memory location; and

restarting execution of the blocked thread.

2. The method of claim 1 wherein when access by the thread to the memory location is blocked,

saving the state of the thread; and

storing a reference to the thread in the memory location.

3. The method of claim 2 wherein the stored reference is a reference to a data structure that identifies the blocked thread and the saved state.

4. The method of claim 3 wherein the data structure indicates the value that was stored in the memory location before storing the reference.

5. The method of claim 1 wherein when access by the thread to the memory location is blocked, storing a reference to the thread in the memory location.

6. The method of claim 1 wherein multiple thread are simultaneously blocked on the memory locations.

7. The method of claim 6 including storing a reference to the blocked threads in the memory location.

8. The method of claim 7 wherein the stored reference is a pointer to a list identifying the blocked threads.

9. The method of claim 8 wherein the list is a linked list.

10. The method of claim 1 wherein when the exception is raised as a result of accessing the memory location, determining whether the access by the blocked thread to the memory location will not block after access by the other thread to the memory location and unblocking the thread when the access will not block.

11. The method of claim 1 wherein when the exception is raised as a result of another thread accessing the memory location, determining whether the access by the other thread will block and, when it will block, blocking execution of that other thread.

12. A method in a computer system for deferring calculation of a value until the value is accessed, the method comprising:

enabling an exception to be raised when a memory location is accessed; and

when an exception is raised as a result of access to that memory location,

calculating the value associated with the memory location; and

providing the calculated value as the result of the access to that memory location.

13. The method of claim 12 wherein the providing of the calculated value includes storing the calculated value in the memory location and performing the access to the memory location to access the calculated value.

14. The method of claim 12 wherein the value is calculated by invoking a routine.

15. The method of claim 14 including storing a reference to the routine in the memory location before the exception is raised.

16. The method of claim 12 including storing a reference to code for calculating the value into the memory location.

17. The method of claim 16 wherein the stored reference identifies parameters for passing to the code.

18. A method in a computer system for detecting access to uninitialized memory, the method comprising:

enabling an exception to be raised when a memory location is accessed; and

when an exception is raised as a result of access to that memory location,

when the access is a write access,

disabling the raising of the exception; and

allowing the write access to the memory location; and

when the access is a read access, indicating that access to uninitialized memory has occurred.

19. The method of claim 18 wherein when speculative loads are enabled, the indicating includes setting a poison bit for a destination register of the read access.

20. A method in a computer system for detecting access to protected memory, the method comprising:

enabling an exception to be raised when a memory location is accessed;

when accessing the memory location with a trap for the exception enabled, detecting and preventing access to that memory location; and

when accessing the memory location with a trap for the exception disabled, allowing access to that memory location.

21. The method of claim 20 wherein when speculative loads are enabled and the access is a read, the indicating includes setting a poison bit for a destination register of the read access.

22. A method in a computer system for accessing a collection of data items, the method comprising:

when adding a data item to the collection,

fetching and adding to a write counter, the fetched write counter pointing to a bucket within a bucket array;

reading from the bucket pointed to by the fetched write pointer using a synchronization access mode of sync;

storing the data item in association with the bucket pointed to by the fetched write pointer;

writing to the bucket pointed to by the fetched write pointer using a synchronization access mode of sync; and

fetching and adding to a lower bound to indicate the number of data items added to the collection.

23. The method of claim 22 wherein the bucket pointed to by the fetched write pointer contains a pointer to a linked list of data items.

24. The method of claim 22 wherein the fetched write pointer modulo a number of buckets in the bucket array points to a bucket within the bucket array.

25. The method of claim 22 wherein the adding adds one to the write counter.

26. The method of claim 22 wherein the adding adds a size of a bucket to the write counter.

27. The method of claim 22 including

when removing a data item from the collection,

fetching and adding to a read counter, the fetched read counter pointing to a bucket within the bucket array;

reading from the bucket pointed to by the fetched read pointer using a synchronization access mode of sync;

removing the data item from association with the bucket pointed to by the fetched read pointer; and

writing to the bucket pointed to by the fetched write pointer using a synchronization access mode of sync.

28. The method of claim 22 including

when removing a data item from the collection,

checking the lower bound to ensure that the collection contains a data item, when it cannot be ensured that the collection contains a data item, indicating that the collection may be empty; and

when it can be ensured that the collection contains a data item,

fetching and adding to a read counter, the fetched read counter pointing to a bucket within the bucket array;

reading from the bucket pointed to by the fetched read pointer using a synchronization access mode of sync;

removing the data item from association with the bucket pointed to by the fetched read pointer; and

writing to the bucket pointed to by the fetched write pointer using a synchronization access mode of sync.

29. The method of claim 28 wherein the checking includes fetching and adding a negative number to the lower bound.

30. The method of claim 29 wherein the checking includes fetching and adding a positive number to the lower bound when it cannot be ensured that the collection contains an item.

31. A method in a computer system for accessing a collection of data items, the method comprising:

when removing a data item from the collection,

fetching and adding to a read counter, the fetched read counter pointing to a bucket within the bucket array;

reading from the bucket pointed to by the fetched read pointer using a synchronization access mode of sync;

removing the data item from association with the bucket pointed to by the fetched read pointer; and

writing to the bucket pointed to by the fetched write pointer using a synchronization access mode of sync.

32. The method of claim 31 including before fetching and adding to the read counter checking a lower bound to ensure that the collection contains a data item.

33. The method of claim 32 wherein it cannot be ensured that the collection contains a data item, indicating that a data item cannot be removed.

34. The method of claim 31 wherein the bucket pointed to by the fetched read pointer contains a pointer to a linked list of data items.

35. The method of claim 31 wherein the fetched read pointer modulo a number of buckets in the bucket array points to a bucket within the bucket array.

36. The method of claim 31 wherein the removing adds one to the read counter.

37. The method of claim 31 wherein the adding adds a size of a bucket to the read counter.

38. A method in a computer system for accessing a buffer of data, the method comprising:

defining a write pointer to point to a location within the buffer;

when adding data to the buffer,

fetching the write pointer;

adding an indication of a size of the data to the write pointer; and

storing the data into the buffer starting at a location indicated by the fetched write pointer; and

setting the synchronization access mode of the write pointer to be either normal or sync to effect the behavior of adding data to the buffer.

39. The method of claim 38 wherein the fetching and adding includes executing a fetch and add operation.

40. The method of claim 38 wherein when the synchronization access mode of the write pointer is set to normal, the storing includes overwriting data previously stored in the buffer and not yet read.

41. The method of claim 38 wherein when the synchronization access mode of the write pointer is set to sync, the storing includes waiting to overwrite data previously stored in the buffer until the data has been read.

42. The method of claim 38 wherein the setting of the synchronization access mode of the write pointer is transparent to the adding of the data to the buffer.

43. A method in a computer system for implementing a circular buffer, the method comprising storing in forwarding words located past an end of the buffer pointers to locations at the other end of the buffer, the pointers having forwarding enabled so that when a forwarding word is accessed, the access is directed to the pointed to word at the other end of the buffer.

44. The method of claim 43 wherein the buffer is pointed to by a write pointer whose value modulo a size of the buffer indicates the starting position for storing data in the buffer.

45. The method of claim 43 wherein the buffer is pointed to by a read pointer whose value modulo a size of the buffer indicates the starting position for reading data from the buffer.

46. A method in a computer system for detecting access to a memory location adjacent to a data structure, the method comprising:

storing a pointer to an invalid memory location in the memory location;

enabling forwarding for the memory location; and

when access to that invalid memory location raises an exception, indicating that the memory location adjacent to the data structure has been accessed.

47. The method of claim 46 wherein when speculative loads are enabled, the indicating includes setting a poison bit in a destination register when the access is a load from that memory location.

48. The method of claim 46 wherein the access does not disable the forwarding.

49. A method in a computer system for observing access to a memory location, the method comprising:

under control of an observed thread, accessing the memory location with a synchronization access mode of normal; and

under control of an observing thread,

accessing the memory location with a synchronization access mode of sync; and

when access to the memory location is successful, indicating access to the memory location by the observed thread.

50. The method of claim 49 wherein the observing thread disables a memory retry-limit trap and when access to the memory location is not successful, re-accessing the memory location with a synchronization access mode of sync.

51. The method of claim 49 wherein when the access to the memory location is not successful, the observing thread is blocked until the observed thread stores a value in the memory location.

52. A method in a computer system of restricting access to memory, the method comprising:

setting a memory location to indicate a trap should occur when the memory location is accessed;

under control of a restricted portion of a computer program,

setting a pointer to point to the memory location that indicates that traps to the pointed to memory location are enabled; and

accessing the memory location using the set pointer so that a trap occurs and access to the restricted memory location is detected; and

under control of an unrestricted portion of the computer program,

setting a pointer to point to the memory location that indicates that traps to the pointed to memory location are disabled; and

accessing the memory location using the set pointer so that a trap does not occur and access to the restricted memory location is allowed.

53. The method of claim 52 wherein a user program typically accesses memory locations using pointers with traps enabled.

54. The method of claim 52 including setting all memory locations of a data structure to indicate a trap should occur when the memory locations are accessed.

55. A computer-readable medium containing instructions for causing a computer system to perform the method of claim 52.

56. A computer system including components to perform the method of claim 52.